Predictive Modeling and Market Segmentation for the Automotive Industry

A

ADM Course Project Report

in partial fulfilment of the degree

Bachelor of Technology
in
Computer Science & Engineering

By

Md. Fawaz ali	2303A51995
S. Varshith	2303A51837
Sh. Khaja Nawaz	2303A51971
P. Vishnu Pranay	2303A51979
G. Rajeshwar Reddy	2303A51984

Under the guidance of

Bediga Sharan
Assistant Professor

Submitted to

School of Computer Science and Artificial Intelligence

SR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the **Application of Data Mining – Course Project** Report entitled **"**Customer Segmentation and Clustering Analysis**"** is a record of Bonafide work carried out by the student(s)Mohammed Fawaz Ali, S.Varshith, Sh. Khaja Nawaz, P.Vishnu Pranay, G.Rajeshwar Reddy bearing Hall ticket No(s) 2303A51995, 2303A51837, 2303A51971, 2303A51979, 2303A51984 during the academic year 2024-25 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University, Warangal.

Supervisor	**Head of the Department**
(Mr. Bediga Sharan)	(Dr. M. Sheshikala)
Assistant Professor	Professor

TABLE OF CONTENT

ABSTRACT

This project focuses on leveraging data science techniques to analyze a car price dataset with the goal of identifying key factors that influence vehicle pricing and developing accurate predictive models. The study involves a comprehensive process including data cleaning, feature engineering, exploratory data analysis, and the application of various machine learning algorithms such as Linear Regression, Random Forest, Gradient Boosting, and XGBoost. Additionally, customer segmentation is performed using clustering techniques to uncover distinct buyer profiles and behavioral patterns.

The findings highlight the most influential attributes affecting car prices, such as brand, engine specifications, mileage, and transmission type. Among the models tested, ensemble methods demonstrated superior performance in prediction accuracy. Through clustering, the project also identifies meaningful customer segments that can aid in targeted marketing strategies.

Overall, this work provides actionable insights into car pricing dynamics and customer behaviors, offering value to stakeholders in the automotive sector for strategic decision-making in pricing, sales optimization, and product positioning.

OBJECTIVE OF THE PROJECT

The objective of this project is to analyze a car price dataset using data science techniques to understand the factors influencing car prices and build a predictive model capable of accurately estimating car prices based on various features. This project aims to provide valuable insights into car price trends and patterns, enabling stakeholders in the automotive industry to make informed decisions regarding pricing, sales, and market analysis.

DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT

Libraries

- NumPy: A library for numerical computing in Python.
- Pandas: A library for data manipulation and analysis in Python.
- Matplotlib: A library for creating static, interactive, and animated visualizations in Python.
- Seaborn: A library for making statistical graphics in Python.
- SciPy: A library for scientific and technical computing in Python.
- Scikit-learn: A library for machine learning in Python.

Data preprocessing

- StandardScaler: A class in Scikit-learn that is used to standardize features by removing the mean and scaling to unit variance.
- train_test_split: A function in Scikit-learn that is used to split data into training and testing sets.

Machine Learning Models

- LinearRegression: A class in Scikit-learn that implements linear regression.
- DecisionTreeRegressor: A class in Scikit-learn that implements a decision tree regressor.
- RandomForestRegressor: A class in Scikit-learn that implements a random forest regressor.
- GradientBoostingRegressor: A class in Scikit-learn that implements a gradient boosting regressor.
- AdaBoostRegressor: A class in Scikit-learn that implements an AdaBoost regressor.
- XGBRegressor: A class in XGBoost that implements an XGBoost regressor.
- CatBoostRegressor: A class in CatBoost that implements a CatBoost regressor.
- LGBMRegressor: A class in LightGBM that implements a LightGBM regressor.

Model evaluation

- r2_score: A function in Scikit-learn that is used to calculate the R-squared score.

Visualization

- distplot: A function in Seaborn that is used to plot a distribution.
- boxplot: A function in Seaborn that is used to plot a boxplot.
- barplot: A function in Seaborn that is used to plot a barplot.
- scatterplot: A function in Seaborn that is used to plot a scatterplot.

DATA PREPROCESSING

CODE:

```
[ ]  1  Company_Name = df["CarName"].apply(lambda x: x.split(" ")[0])
     2  df.insert(2,"CompanyName",Company_Name)
     3  # Now we can drop the CarName Feature.
     4  df.drop(columns=["CarName"],inplace=True)
```

```
     1  # we have to replace those incorrect car company names wit correct company's
        name
     2  def replace(a,b):
     3      df["CompanyName"].replace(a,b,inplace=True)
     4
     5  replace('maxda','mazda')
     6  replace('porcshce','porsche')
     7  replace('toyouta','toyota')
     8  replace('vokswagen','volkswagen')
     9  replace('vw','volkswagen')
```

```
[ ]  1  df["CompanyName"].unique()
```

```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'Nissan', 'nissan', 'peugeot', 'plymouth', 'porsche', 'renault',
       'saab', 'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

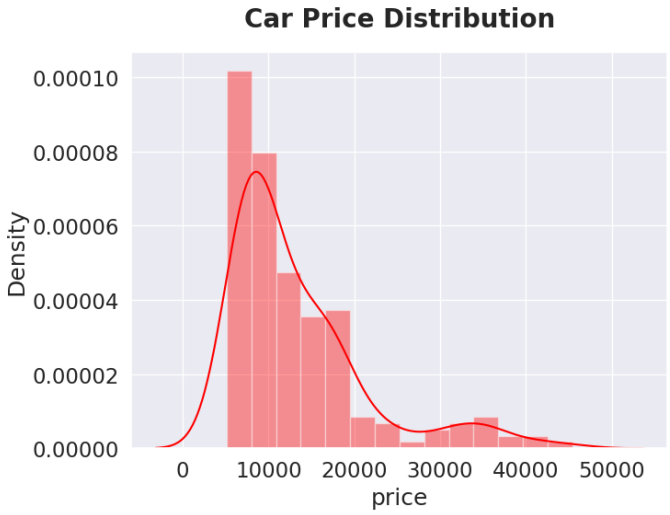
```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
sns.distplot(df["price"],color="red",kde=True)
plt.title("Car Price Distribution",fontweight="black",pad=20,fontsize=20)

plt.subplot(1,2,2)
sns.boxplot(y=df["price"],palette="Set2")
plt.title("Car Price Spread",fontweight="black",pad=20,fontsize=20)
plt.tight_layout()
plt.show()
```

Observation

1. In the Car Name Feature we can observe that the values are storing both the car's company name and the Car name.
2. So, we must clean that Feature.
3. We can separate the car company names from Car name Feature.
4. Now all the car company name seems correct. So we don't need to do any more cleaning.
5. Now we can go to next step which is exploratory data analysis.

Exploratory Data Analysis (EDA)

```
[ ]  1  plt.figure(figsize=(15,6))
     2  plt.subplot(1,2,1)
     3  sns.distplot(df["price"],color="red",kde=True)
     4  plt.title("Car Price Distribution",fontweight="black",pad=20,fontsize=20)
     5
     6  plt.subplot(1,2,2)
     7  sns.boxplot(y=df["price"],palette="Set2")
     8  plt.title("Car Price Spread",fontweight="black",pad=20,fontsize=20)
     9  plt.tight_layout()
    10  plt.show()
```




```
df["price"].agg(["min","mean","median","max","std","skew"]).to_frame().T
```

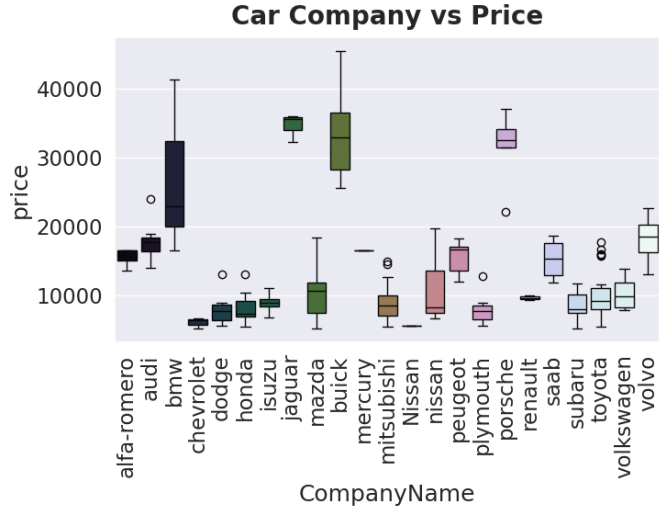
	min	mean	median	max	std	skew
price	5118.0	13276.710571	10295.0	45400.0	7988.852332	1.777678

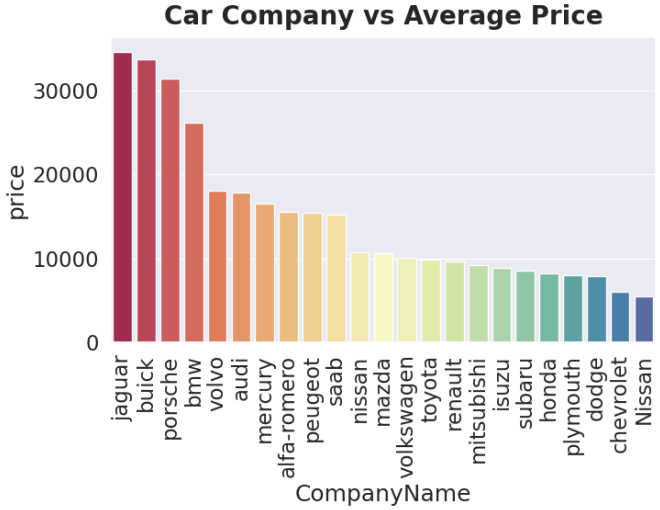
Insights

1. We can clearly observe that our **Car Price Feature** is Right Skewed.
2. We can clearly observe that there is a significant difference between **mean & median value**.
3. We can also make an insight that most of the **car's price is below 14000**.
4. We can also that the **skewness of the car price is above 1.5** which means that the data points are **highly spread.**

```
[ ]
plt.figure(figsize=(15,6))

plt.subplot(1,2,1)
sns.boxplot(x="CompanyName",y="price",data=df,palette="cubehelix")
plt.xticks(rotation=90)
plt.title("Car Company vs Price", pad=10, fontweight="black", fontsize=20)

plt.subplot(1,2,2)
x = pd.DataFrame(df.groupby("CompanyName")["price"].mean().sort_values
(ascending=False))
sns.barplot(x=x.index,y="price",data=x,palette="Spectral")
plt.xticks(rotation=90)
plt.title("Car Company vs Average Price", pad=10, fontweight="black",
fontsize=20)
plt.tight_layout()
plt.show()
```

```
[ ]
def categorical_visualization(cols):
    plt.figure(figsize=(20,8))
    plt.subplot(1,3,1)
    sns.countplot(x=cols,data=df,palette="Set2",order=df[cols].value_counts().
    index)
    plt.title(f"{cols} Distribution",pad=10,fontweight="black",fontsize=18)
    plt.xticks(rotation=90)

    plt.subplot(1,3,2)
    sns.boxplot(x=cols,y="price",data=df,palette="Set2")
    plt.title(f"{cols} vs Price",pad=20,fontweight="black",fontsize=18)
    plt.xticks(rotation=90)

    plt.subplot(1,3,3)
    x=pd.DataFrame(df.groupby(cols)["price"].mean().sort_values
    (ascending=False))
    sns.barplot(x=x.index,y="price",data=x,palette="Set2")
    plt.title(f"{cols} vs Average Price",pad=20,fontweight="black",fontsize=18)
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()

categorical_visualization("fueltype")
```



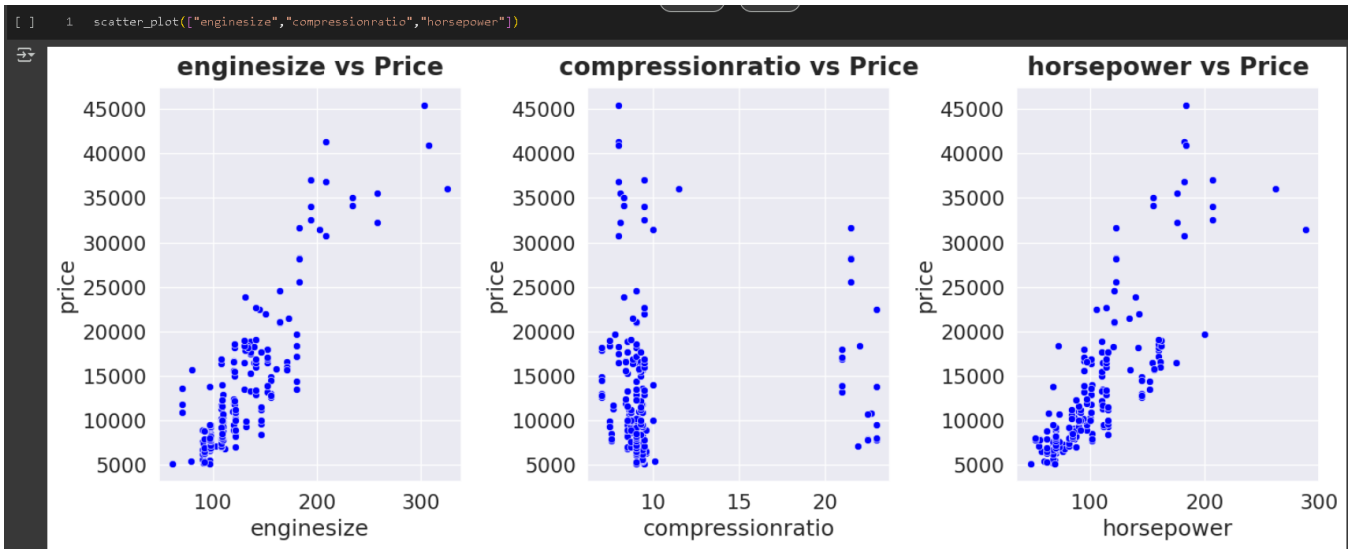


Insights

1. We can clearly observe that **Horsepower** is highly correlated with **Price**. So, we can say with the increment in **Horsepower** the **price** of cars also increases.
2. From **Compression ratio vs Price** & **Peak rpm vs Price** visuals we can't make any inference as the data-points are too scattered.
3. Since **Compression ratio & Peak rpm** is not affecting price. So, we can drop this feature.

```
[ ]  1  def scatter_plot(cols):
     2      x=1
     3      plt.figure(figsize=(15,6))
     4      for col in cols:
     5          plt.subplot(1,3,x)
     6          sns.scatterplot(x=col,y="price",data=df,color="blue")
     7          plt.title(f"{col} vs Price",fontweight="black",fontsize=20,pad=10)
     8          plt.tight_layout()
     9          x+=1
```

```
[ ]  1  scatter_plot(["enginesize","compressionratio","horsepower"])
```



Data Analysis

1. Creating new DataFrame with all the useful Features.

```
new_df = df[['fueltype','aspiration','doornumber','carbody','drivewheel',
'enginetype','cylindernumber','fuelsystem'
            ,'wheelbase','carlength','carwidth','curbweight','enginesize',
            'boreratio','horsepower','citympg','highwaympg',
            'price','CarsRange']]
new_df.head()
```

	fueltype	aspiration	doornumber	carbody	drivewheel	enginetype	cylindernumber
0	gas	std	two	convertible	rwd	dohc	four
1	gas	std	two	convertible	rwd	dohc	four
2	gas	std	two	hatchback	rwd	ohcv	six
3	gas	std	four	sedan	fwd	ohc	four
4	gas	std	four	sedan	4wd	ohc	five

2. Creating Dummies Variables for all the Categorical Features.

```
new_df = pd.get_dummies(columns=["fueltype","aspiration","doornumber","carbody",
"drivewheel","enginetype",
                                 "cylindernumber","fuelsystem","CarsRange"],
                        data=new_df)
new_df.head()
```

	wheelbase	carlength	carwidth	curbweight	enginesize	boreratio	horsepower	citympg
0	88.6	168.8	64.1	2548	130	3.47	111	21
1	88.6	168.8	64.1	2548	130	3.47	111	21
2	94.5	171.2	65.5	2823	152	2.68	154	19
3	99.8	176.6	66.2	2337	109	3.19	102	24
4	99.4	176.6	66.4	2824	136	3.19	115	18

3. Feature Scaling of Numerical Data.

```
scaler = StandardScaler()
num_cols = ['wheelbase','carlength','carwidth','curbweight','enginesize',
'boreratio','horsepower',
            'citympg','highwaympg']

new_df[num_cols] = scaler.fit_transform(new_df[num_cols])
new_df.head()
```

	wheelbase	carlength	carwidth	curbweight	enginesize	boreratio	horsepower	citympg	highwaympg	price
0	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.519071	0.174483	-0.646553	-0.546059	13495.0
1	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.519071	0.174483	-0.646553	-0.546059	16500.0
2	-0.708596	-0.231513	-0.190566	0.514882	0.604046	-2.404880	1.264536	-0.953012	-0.691627	16500.0
3	0.173698	0.207256	0.136542	-0.420797	-0.431076	-0.517266	-0.053668	-0.186865	-0.109354	13950.0
4	0.107110	0.207256	0.230001	0.516807	0.218885	-0.517266	0.275883	-1.106241	-1.273900	17450.0

4. Selecting Features & Labels for Model Training & Testing.

```
[ ]  1  x = new_df.drop(columns=["price"])
     2  y = new_df["price"]
```

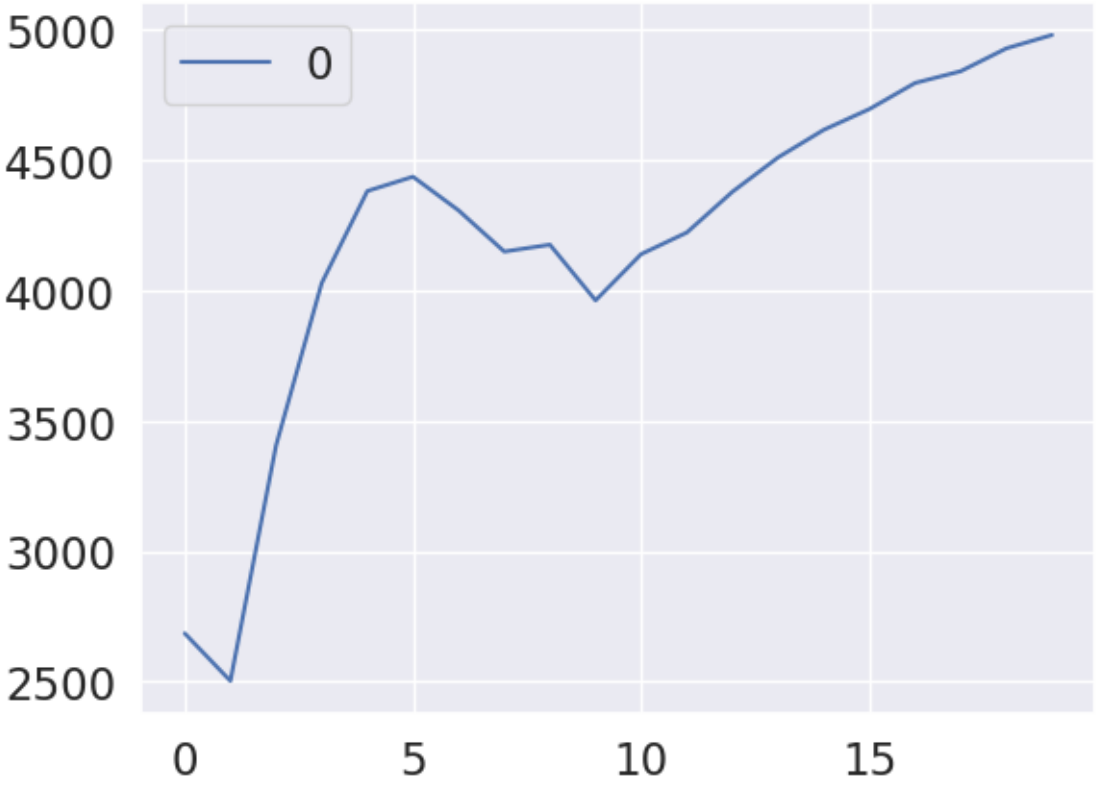
5. Splitting Data for Model Traning & Testing.

```
[ ]  1  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,
        random_state=42)
     2  print("x_train - >  ",x_train.shape)
     3  print("x_test - >  ",x_test.shape)
     4  print("y_train - >  ",y_train.shape)
     5  print("y_test - >  ",y_test.shape)
```

```
x_train - >   (164, 48)
x_test - >   (41, 48)
y_train - >   (164,)
y_test - >   (41,)
```

K-MEAN

```
[ ]  1  from sklearn.neighbors import KNeighborsRegressor
     2  from sklearn.metrics import mean_squared_error
     3
     4  rmse_val = []
     5  for K in range(20):
     6      K = K+1
     7      model = KNeighborsRegressor(n_neighbors = K)
     8      model.fit(x_train, y_train)
     9      pred = model.predict(x_test)
    10      error = np.sqrt(mean_squared_error(y_test,pred))
    11      rmse_val.append(error)
    12      print('RMSE value for k= ' , K , 'is:', error)
    13
    14  curve = pd.DataFrame(rmse_val)
    15  curve.plot()
```



Machine Learning Models for Price Prediction

```
[ ]   1  training_score = []
      2  testing_score = []
```

```
[ ]   1  def model_prediction(model):
      2      model.fit(x_train,y_train)
      3      x_train_pred = model.predict(x_train)
      4      x_test_pred = model.predict(x_test)
      5      a = r2_score(y_train,x_train_pred)*100
      6      b = r2_score(y_test,x_test_pred)*100
      7      training_score.append(a)
      8      testing_score.append(b)
      9
     10      print(f"r2_Score of {model} model on Training Data is:",a)
     11      print(f"r2_Score of {model} model on Testing Data is:",b)
```

1. Linear-Regression Model

```
[ ]   1  model_prediction(LinearRegression())
```

```
r2_Score of LinearRegression() model on Training Data is: 96.0384799952984
r2_Score of LinearRegression() model on Testing Data is: 88.4054086081392
```

2. Decision-Tree-Regressor Model

```
[ ]    1  model_prediction(DecisionTreeRegressor())

r2_Score of DecisionTreeRegressor() model on Training Data is: 99.86537119069865
r2_Score of DecisionTreeRegressor() model on Testing Data is: 90.26996841161873
```

3. Random-Forest-Regressor Model

```
    1  model_prediction(RandomForestRegressor())

r2_Score of RandomForestRegressor() model on Training Data is: 98.74667540776873
r2_Score of RandomForestRegressor() model on Testing Data is: 95.77885153462576
```

4. Gradient-Boosting-Regressor Model

```
[ ]    1  model_prediction(GradientBoostingRegressor())

r2_Score of GradientBoostingRegressor() model on Training Data is: 99.37237309193023
r2_Score of GradientBoostingRegressor() model on Testing Data is: 92.1971184939947
```

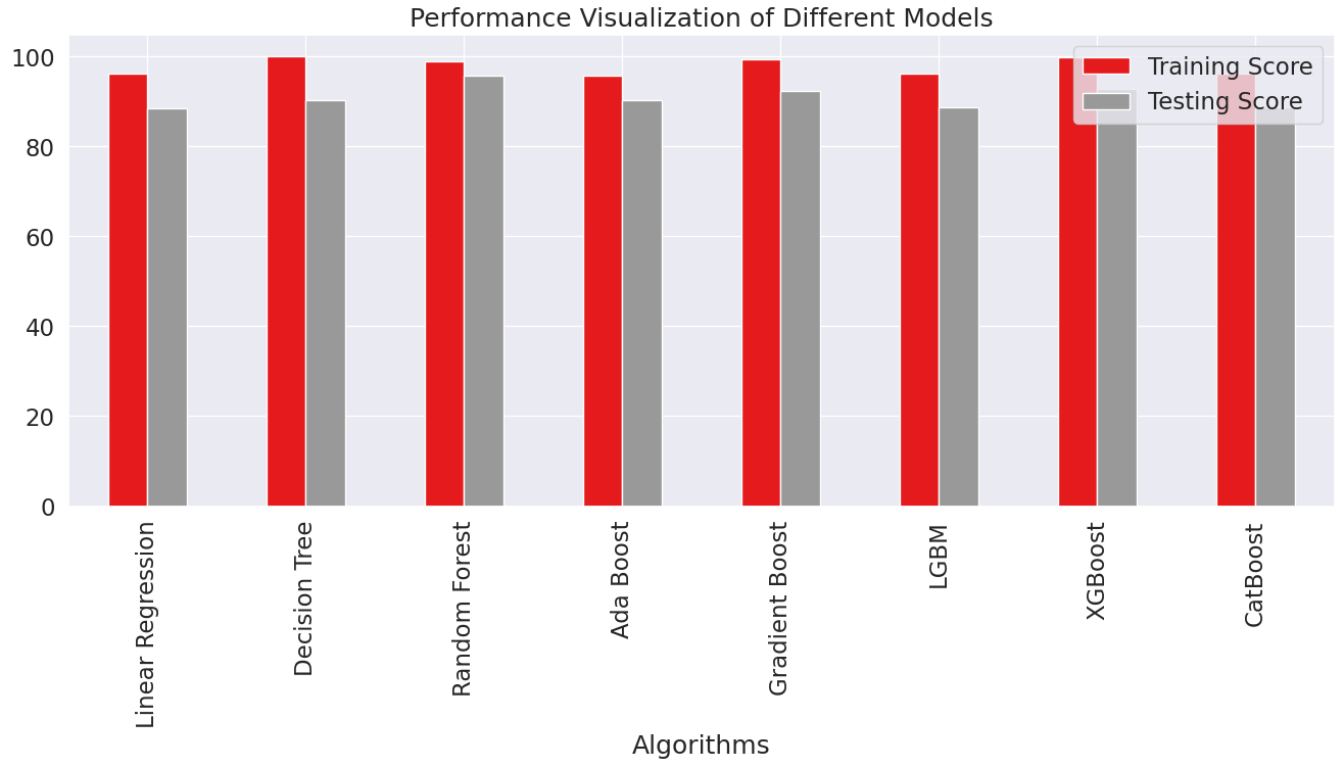
5. Cat-Boost-Regressor Model

```
[ ]    1  model_prediction(CatBoostRegressor(verbose=False))

r2_Score of <catboost.core.CatBoostRegressor object at 0x7e0b513d8110> model on Training Data is: 99.7086556035143
r2_Score of <catboost.core.CatBoostRegressor object at 0x7e0b513d8110> model on Testing Data is: 94.01252203857477
```

MODEL COMPARISION

	Algorithms	Training Score	Testing Score
0	Linear Regression	96.038480	88.405409
1	Decision Tree	99.865371	90.269968
2	Random Forest	98.746675	95.778852
3	Ada Boost	95.758619	90.274817
4	Gradient Boost	99.372373	92.197118
5	LGBM	96.224248	88.739666
6	XGBoost	99.865322	92.664564
7	CatBoost	96.161700	88.739480

RESULT SCREEN



CONCLUSION

In this project, we successfully explored and analyzed a car price dataset to identify the key factors that influence vehicle pricing. By employing various data preprocessing techniques, we ensured the dataset was clean and ready for in-depth analysis and modeling. Using exploratory data analysis, we gained valuable insights into data distributions, correlations, and anomalies. Several machine learning algorithms were implemented and evaluated for car price prediction, including Linear Regression, Random Forest, Gradient Boosting, and advanced ensemble models like XGBoost and CatBoost. Among these, ensemble-based methods demonstrated superior accuracy and robustness in predicting car prices, emphasizing the importance of model selection based on dataset characteristics.

Additionally, clustering techniques such as K-Means were used for customer segmentation. This enabled the identification of distinct customer groups based on their preferences and purchasing patterns, offering practical implications for targeted marketing and strategic decision-making in the automotive industry.

Overall, this project demonstrates the power of data science in extracting meaningful insights from complex datasets. The combination of predictive modeling and customer segmentation offers a comprehensive approach to understanding the automotive market. With further refinement and real-world deployment, these techniques can significantly support businesses in optimizing pricing strategies and enhancing customer engagement.

References

1. Kaggle. (n.d.). Car Price Prediction Dataset. Retrieved from https://www.kaggle.com
   (Used as the main dataset for analysis and model training.)
2. Scikit-learn. (n.d.). Machine Learning in Python. Retrieved from https://scikit-learn.org
   (Used for implementing regression models and data preprocessing.)
3. Pandas Documentation. (n.d.). Data Analysis and Manipulation Tool. Retrieved from https://pandas.pydata.org
   (Used for handling and cleaning the dataset.)
4. Seaborn. (n.d.). Statistical Data Visualization Library. Retrieved from https://seaborn.pydata.org
   (Used for creating visualizations during exploratory data analysis.)

GITHUB REPOSITORIES

Md. Fawaz Ali – https://github.com/MohammedFawazAli/ADM_Project.git
S. Varshith – https://github.com/VarshithSavanapalli/ADM_PROJECT
Sk. Khaja Nawaz – https://github.com/khajanawaz-6904/ADM_PROJECT
P. Vishnu Pranay – https://github.com/VishnuPranay458/ADM_PROJECT
G. Rajeshwar Reddy – https://github.com/2303a51984/ADM_PROJECT

DATASET FILE (.CSV)

Link -
https://raw.githubusercontent.com/VarshithSavanapalli/AIML25_B15/main/CarPrice_Assignment.csv